A Unified Architecture for Surfacing the Content of Deep Web Databases

G. Pavai[1], Dr. T. V. Geetha[2]
[1]Anna University, Chennai, India
Email: pavai_gops@yahoo.co.in
[2]Anna University, Chennai, India
Email: tv_g@hotmail.com

***Abstract*—Deep web is the part of web that is not technically accessible to a search engine's crawler. The Deep web contains more than five hundred times the data present in the surface web that search engines normally search. But we don't have access to these data directly. In this paper, we propose a unified architecture for accessing the data from Deep web databases. We have also used the heuristic of using the data present for input restricted fields in the forms for automatically filling the unified query interface to extract the data from the deep web databases. We have evaluated our work using the precision metric by considering data from ten Deep web databases. In future, we have plans to increase the number of databases from which we extract the deep web data and also to increase the range of queries that are to be answered by extracting data from the Deep web databases. We are also working on tracking changes in deep web databases using an incremental crawl strategy**.

***Index Terms*—Deep web, surfacing, query interface, forms, data extraction**

I. Introduction

Li et al. [7] say that the deep Web is 400 to 500 times larger than the commonly defined World Wide Web, and is growing at a higher rate. Data in Deep Web grows exponentially and covers great subject diversity apart from providing authoritative information. It is this size of the deep web, its exponential growth with time and the authoritative information that it provides us with, has made deep web one of the hot areas of research in the field of Information Retrieval

To discover content on the Web, search engines use web crawlers that follow hyperlinks through known protocol virtual port numbers. This technique is ideal for discovering resources on the surface web but is often ineffective at finding deep Web resources. For example, these crawlers do not attempt to find dynamic pages that are the result of database queries due to the infinite number of queries that are possible. It has been noted that this can be (to a certain extent) overcome by providing links to query results, but this could unintentionally inflate the popularity for a member of the deep Web. Research has proved that over 95% of the web data is hidden in the web databases which are in no way accessible to the surface web user. Normally, a person gets access to these data via filling details in the query interfaces (forms) and submitting them to the server. The server then sends queries to the web databases and retrieves results as per the details filled in by the user in the form. Such result pages are normally having dynamic URLs. Such dynamic URLs cannot be indexed by the search engines. So we lack access to such abundant data because it is hidden in the deep web. Currently, researchers are working on bringing out the deep web data for the search engine users.

There are two methodologies for accessing the deep web data. One is, the user gives a query, and accordingly, the data from the query is used to fill a relevant query interface and it is posted to the web server. The web server accesses the data in the web database by sending a query to the web database. The result page of the database access is showed as the required result page. The other methodology is surfacing the hidden web data i.e. we retrieve all possible data from a given web database by giving queries to the web database and extracting the data from the result pages. The data thus extracted is checked for duplicates and the data is written to an html file thereby giving the result page a static URL. This page now behaves just like any other web page i.e. can be indexed. So, now it is a searchable document. We follow this approach since the former consumes online search time. Whereas, in the surfacing of the deep web content, since the data extraction is done offline, the search time does not increase. In this paper, we propose a unified architecture for surfacing data from domain specific deep web databases.

II. Background

Bin He et al. [6] discusses about issues present in Deep web like, where to find the required domain specific query interfaces and to what depth should the crawl be performed to get the query interfaces. the scale of the Deep web, Deep web is structured or not, the subjects contained in the Deep web databases, how far popular search engines handle Deep web data, and the coverage of Deep web directories.Ying Wang et al. [8] discusses how to discover domain specific Deep web sources by using a focussed crawling methodology that locates web sites for domain specific data sources and then judges whether the corresponding Deep web query interface exists within three stages of crawl and then checking whether the query interface is relevant to a particular topic. While this paper deals with how domain specific Deep web query interfaces are found out, Hui Li et al. [7] discuss an effective incremental web crawler that maintains an up to date local repository of web pages by calculating priorities for all the web pages. We do not concentrate on the Deep web source discovery in our work. We have manually selected ten online

DOI: 03.LSCS.2013.1.82

databases, under the tourism domain, whose contents are not normally visible to the surface search engine users.Bin He et al. [4] discuss how web query interfaces are integrated using a three step process, where step 1 is interface extraction process where the attributes in the query interface are automatically extracted, step 2 is schema matching where semantic correspondences among attributes are found out and step 3 is interface unification where we construct a unified query interface based on the matches we find in step 2. The same process of web query unification is being dealt with in a different method in Pierre et al. [5] where domain knowledge is used to unify the query interfaces. In our work, we combine both the above methodologies to form the unified query interface.

Pierre et al. [5] also discuss how to extract the valuable data from Deep web databases by probing the forms with all types of values and analysing all the result pages to obtain all the fields of the Deep web database. Then many such result pages are analysed and maximum amount of data is extracted from the web databases by probing the query interfaces with all possible values to all the fields. In our work, we have used the auto-filling field's data that is available in the forms to fill the query interfaces in order to extract data from the deep web databases.

Jufeng et al. [1] proposes a technique for data extraction from hidden web pages using rules of structure, logic and application whereas, Anuradha and A.K. Sharma. [2] Proposes a technique for data extraction from hidden web databases by identifying the templates and tag structures of a document by sending queries through HTML query interfaces of the Databases. But, no single database will provide all desired data. This gives rise to F. Wang et al. [3] that takes a further step by extracting data from multiple databases. Some issues handled in their work are the order in which the multiple Databases are queried before the others and the non-availability of certain databases at times due to network and hardware problems that is handled in this work by proper query planning to design alternative plans when optimal plans fail. Our work handles the multiple database problems by just removing the duplicates. We do not handle the query plans part.

III. Methodology

Normally, we say that the content of deep web databases is not directly visible to the surface search engine users. But, the query interfaces that are indexed to the search engine are displayed in the results for a user query based on the relevance of the form to the query. If the user happens to click on this link and fill in the form and submit it to the web server, the results of the query to the database may be displayed. This is one way of getting the deep web data using a surface search engine. But it depends on the probability of the user clicking on the query interface. This is an online process, so the time to get the results after giving the user gives a query is high. So, we go in for doing the filling in of forms and extraction of data from the result pages as an offline process and index the extracted results thereby reducing the time to get the desired results. Fig 1 gives a detailed description of the deep web architecture.

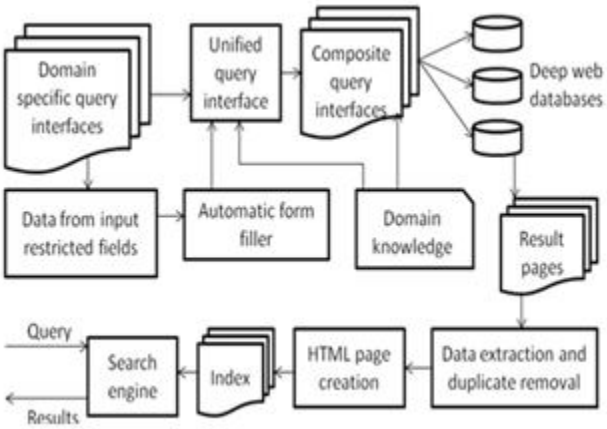

Fig. 1. Deep Web Architecture

A. Unification of Query Interfaces

Finding out domain specific query interfaces is the first step in deep web search. Deep web search is a very specific search process and cannot be generalized. So, finding out only the domain specific query interfaces is very important. This is a focused research topic in the Deep web domain. But we manually find the required query interfaces for our work because this is beyond the scope of our work as we would like to concentrate on surfacing the deep web content and not in identifying the domain specific deep web databases. But still, for the accuracy of the results, we manually found out the required query interfaces. The domain that we used for our work is the tourism domain. The query interfaces that we use for our work are the online air ticket reservation forms from ten different web sites. For unifying these forms together to form a unified query interface, we have used domain knowledge [5].

Domain knowledge based query interface unification is done here so as to resolve problems where one website may call the source of journey as "from" and another may call it as "start". For example, Cleartrip.com considers the "source" and "destination" fields as "from" and "to", makemytrip.com considers them as "leaving from" and "going to" and goindigo.com considers them as "origin" and "destination". Though, this can be easily identified by human beings, machines cannot do this. Figure 2, shows the query interface of cleartrip.com and figure 3, Shows the query interface of makemytrip.com. The source field of both the query interfaces is highlighted in these two figures. We have showed our unified query interface in Figure 4. We have highlighted the source field in the unified query interface also. Our unified query interface gives a common name for the same fields, with different names in the different query interface.While unifying two or more query interfaces, we need to overcome such problems and so, we are giving the domain knowledge as input to the query unification process. The new unified query interface incrementally adds all the fields present in all of the component query interfaces [4] and similar fields are not repeated.

DOI: 03.LSCS.2013.1.82

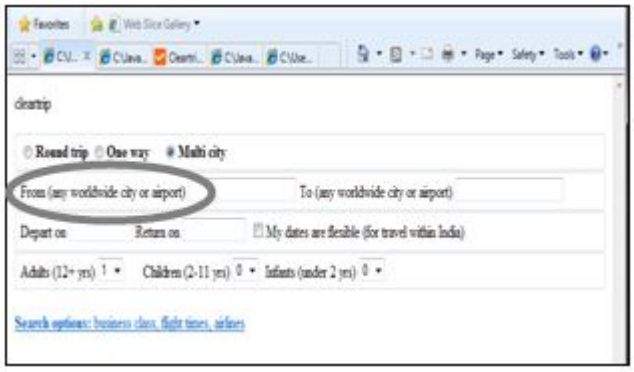

Fig. 2. Cleartrip.com's query interface

Fig. 3. Makemytrip.com's query interface

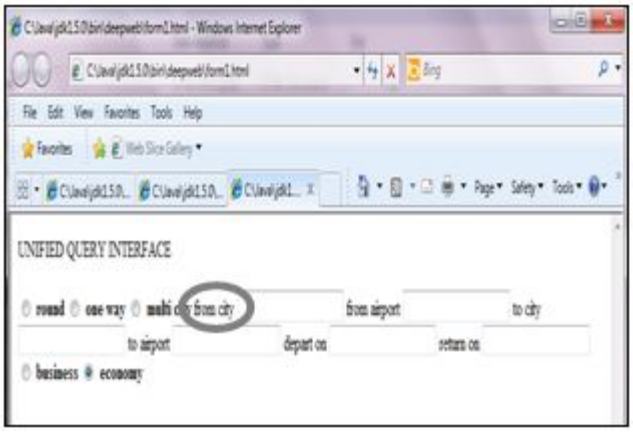

Fig. 4. The unified query interface

B. Automatic Form Filling and Data Extraction

Once the unified query interface is ready, we automatically fill it in with the details from the input restricted fields of the composite query interfaces. We also use the heuristic of filling only the mandatory fields rather than filling all the fields [2] along with our input restricted fields heuristics. One example could be the source cities and the destination cities in the forms i.e.only the names of Indian cities are allowed to be filled in the source and destination columns of the form in goindigo.com. The field values are in turn filled into the composite query interfaces and the query is submitted to the respective web databases. The reverse mapping of the fields from the unified query interface to the composite query interfaces is also done with the help of domain knowledge. The result pages obtained by the above process are saved. Then we use the data extraction methodology where data in between tags<TD> and </TD> will be giving the required data [2]. The data so obtained is checked for any duplicates and are stored in a different file for analysis to be done. We analyse the results to get the results for certain queries that have the answers in the databases that we have considered. Some such queries are, 1) maximum airfare cost for an adult from place A to place B 2) minimum airfare cost for an adult from place A to place B 3) maximum airfare cost for a child from place A to place B 4) minimum airfare cost for a child from place A to place B 5) Name of the flight providing the cheapest service 6) number of tickets available from place A to place B on a given date. The results of the analysis are stored in the form of HTML files.

C. Indexing and Searching

The HTML files created as a result of deep web data extraction are indexed to the search engine just like any other crawled page is being indexed. Once the pages are indexed, we can go in for the search process. The important point to be noted here is that the present approach cannot answer all type of queries. Queries based on the pages that we have indexed alone are answerable. Once the query is given, search process is like the normal search process and the retrieved results are displayed. We can use any content based ranking methodology here. But we found that the link analysis based algorithms are not efficient in ranking the surfaced deep web pages since, these are a kind of stand-alone pages that do not have any links thereby giving the minimum rank to our pages that contain the data extracted from the deep web databases.

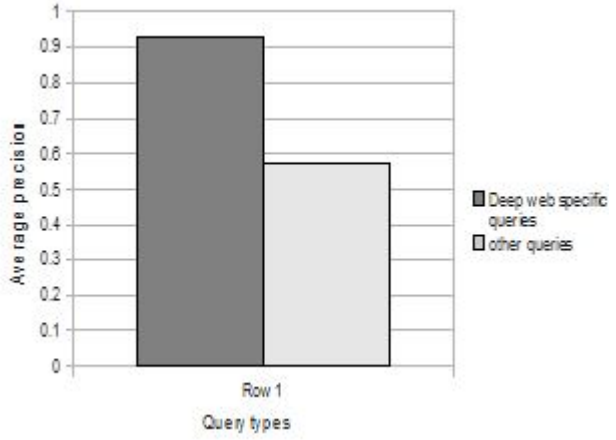

Fig. 5. Average Precision

IV. Evaluation

We have considered 15000 tourism specific documents along with the deep web content extracted to 7 HTML files. We gave the search engine 20 queries of which 12 were deep web specific queries (of the form given in section 3.2) and the other queries were not related to the content in the databases that we have considered. Any search engine can be used for this purpose. But a Link analysis based algorithm will not be fruitful for ranking the surfaced Deep web content since these pages neither contain in-links nor contain out-links.So,

DOI: 03.LSCS.2013.1.82

content based ranking algorithms can be used. Figure 5 gives the average precision for the deep web specific queries and other queries. Our experimental evaluation shows that the deep web specific queries had an average precision of 0.93 which is a good value. The other queries, that did not have any relevant information in the deep web databases that we considered for our work, had to search for the answers in the index of the other crawled pages. The average precision for these queries is only 0.56.

Though the difference in the average precision values between these two types of queries stand as a strong evidence of the presence of more relevant data in the deep web databases, we should remember that not all the Deep web data can be surfaced and accessed via the surface search. If this has to be done all possible set of values should be filled in the unified query interface, the result pages saved, data extracted from them, integrate all such result pages after removing the duplicate entries, create HTML pages for the data and index them to the search engine. But this is a tedious process to do. This search that we considered so far is domain specific (tourism). There are several domains and several deep web databases for each domain which adds to the space and analysis time complexity of indexing all the deep web content to the surface search engine. Another important fact to be considered is that certain data in the Deep web databases keeps on changing instantaneously. For example, the number of available tickets in a flight on a specific trip could have been 20 when the data was extracted from the deep web database but it would have changed to just 2 within a few minutes. But the user gets the result as 20, till the next time the data extraction is done. This is a major drawback of this method. There is no use of indexing such results since by the time the data page is indexed to the search engine, it becomes outdated. Such data can be dealt with dynamically accessing the Deep web databases at the cost of time. We are currently working on an incremental crawl strategy that keeps track of the changes in the databases, so that the above said problem is being dealt with efficiently.

Conclusion & Future Works

In this work, we have proposed a unified architecture for surfacing deep web data from deep web databases to surface search engines. We have also proposed the use of the data for input restricted fields of the forms for the automatic form filling purpose. We have also proved that deep web contains more relevant data for certain user queries through our evaluation. As future works, we would like to work on improving the quantity of the deep web content on the surface web and also keep track of the changes in databases effectively, so as to provide the correct results with respect to time for a given user query.

References

[1] Jufeng Yang, Guangshun Shi Yan and Zheng Qingren Wang, "Data Extraction from Deep Web Pages", International Conference on Computational Intelligence and Security, 2007.

[2] Anuradha, A.K.Sharma, "A Novel Technique for Data Extraction from Hidden Web Databases", International Journal of Computer Applications (0975 – 8887)Volume 15, No.4, February 2011.

[3] F. Wang, G. Agrawal, and R. Jin, "Query Planning for Searching Inter-dependent Deep-Web Databases", in Proc. SSDBM, 2008, pp.24-41.

[4] Bin He, Zhen Zhang, Kevin Chen-Chang, " Knocking The Door To The Deep Web: Integrating Web Query Interfaces", in Proc . ACM- SIGMOID, 2004.

[5] Pierre Senellart, Avin Mittal, Daniel Muschick, Remi Gilleron, Marc Tommasi, " Automatic Wrapper Induction From Hidden-Web Sources with Domain Knowledge", in Proc. ACM – WIDM, 2008.

[6] Bin He, Mitesh Patel, Zhen Zhang, Kevin Chen-Chang, "Accessing The Deep Web: A Survey". Communications of the ACM, v.50 n.5, p.94-101, 2007.

[7] Hui Li, Mei Guo, Liang Cai, Yanwu Yang, " An Incremental Update Strategy in Deep Web", Sixth International conference on Natural Computation (ICNC 2010).

[8] Yin Wang, Wanli Zuo, Tao Peng, Fengling He, "Domain Specific Deep Web Sources Discovery", Fourth International Conference on Natural Computation – IEEE, 2008.

DOI: 03.LSCS.2013.1.82